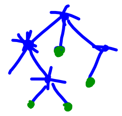
Algebra/algebraic Semantics:

-> term structure

=> tree structure -> rather simple/ easy to understand



atomic - basic expressions

-> operators <- define semantics of operators individually!

=> define the semantics of the whole term by STRUCTURAL INDUCTION

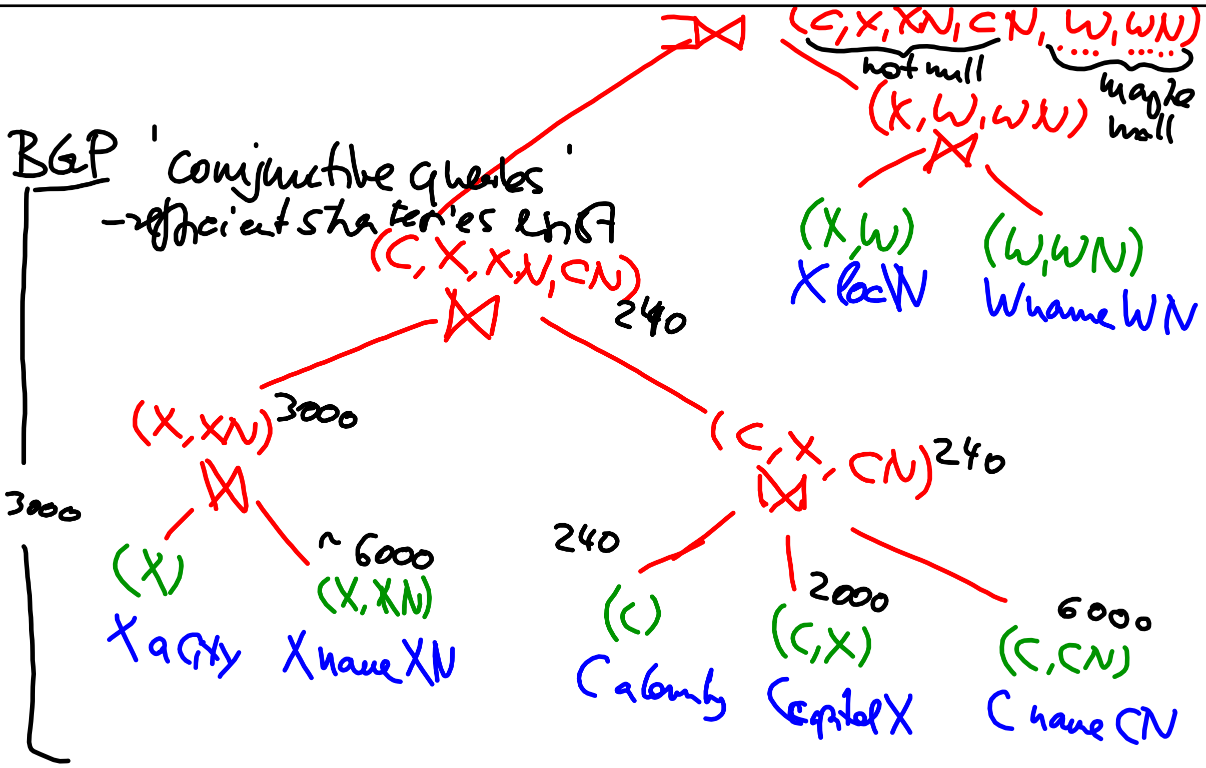
modular, extensible "language"

Rel. Algebra: Rel. Names <- Atomic

Operators: σ, π, ⋈, ∩, ∪, \, ⟕, ⟗, ⟖, ÷, ⋉, additional ops: anti-join, γ <- group by, ...

Mai 22-10:10

BGP 'conjunctive queries' - efficient strategies exist



Mai 22-10:28

?P :mother ∘ :father ?Q

≡ ?P :mother ?X . ?X father ?Q

?P :father* ?Q

→ transitive closure

Mai 22-10:46

A

X	Y
a	null
b	1

B

Y	Z
1	3
1	5
null	4
null	6
2	7

SQL

$\bowtie_{A_2 = B_1}$

A_1	A_2	B_1	B_2
b	1	1	3
b	1	1	5

nothing else

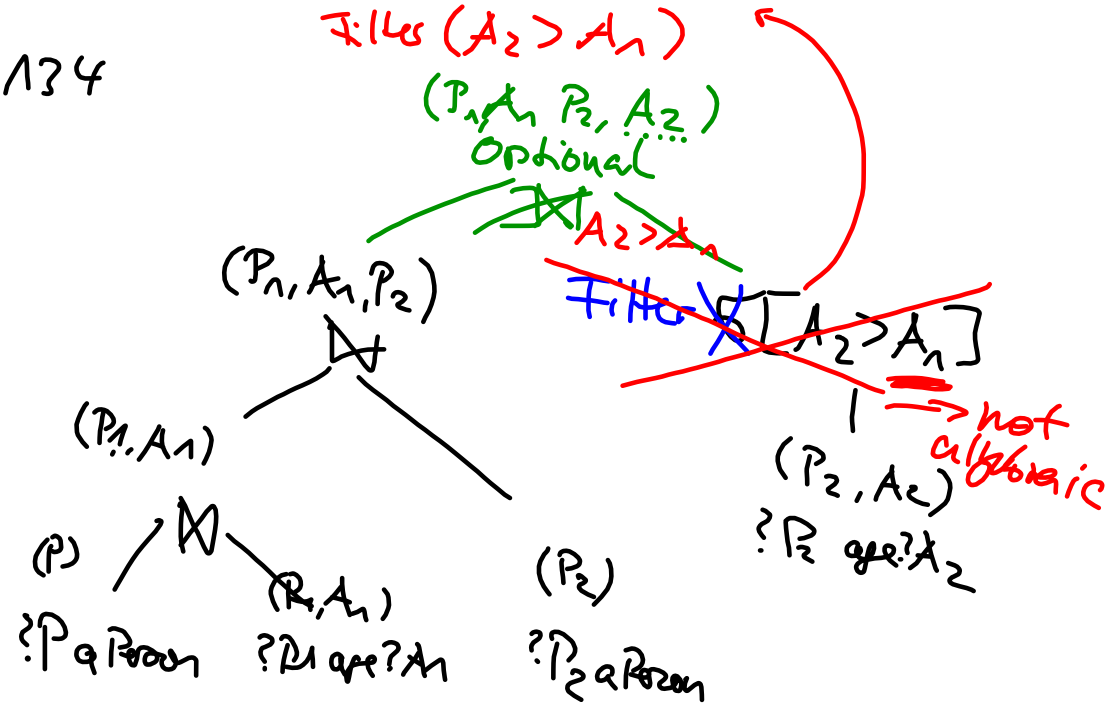
SPARQL:

X	Y	Z
a	1	3
a	1	5
a	null	4
a	null	6
b	1	3
b	1	5
b	1	4
b	1	6

a 2 7

Mai 22-10:58

Se 134



Mai 22-11:10

$\Omega_1 \setminus \Omega_2 = \ldots\ldots$

Consider

$\Omega_1 \ltimes \Omega_2$ "left semijoin"

$= \mu \in \Omega_1$ s.t. exists $\mu_2 \in \Omega_2$ s.t. μ and μ_2 are compatible

in SQL $\equiv$ Select *
from Q1
where exists (Select *
from Q2
where .....)

$= \pi_{[\text{attrs of } \Omega_1]}(\Omega_1 \bowtie \Omega_2)$

Mai 22-11:28

Consider now again

$$\Omega_1 \setminus \Omega_2$$

"anti-join"

$$= \Omega_1 - (\Omega_1 \ltimes \Omega_2)$$

$\hat{=}$ select *
from Q_1
where NOT EXISTS (select *
from Q_2
where....)

Mai 22-11:32

$\Rightarrow$ from this:

$$\Omega_1 ⟕ \Omega_2 = \Omega_1 \bowtie \Omega_2 \cup \Omega_1 \setminus \Omega_2$$

Full outer join:

$$\Omega_1 ⟗ \Omega_2 = \Omega_1 \bowtie \Omega_2 \cup \Omega_1 \setminus \Omega_2 \cup \Omega_2 \setminus \Omega_1$$

Mai 22-11:33